Real-time complexity constrained encoding

T. Vermeir, J. Slowack, G. Van Wallendael, P. Lambert, and R. Van de Walle

In: 2016 IEEE International Conference on Image Processing (ICIP), 819-823, 2016.

REAL-TIME COMPLEXITY CONSTRAINED ENCODING

Thijs Vermeir[⋆†], *Jürgen Slowack*[⋆], *Glenn Van Wallendael*[†], *Peter Lambert*[†], *Rik Van de Walle*[†]

[⋆]Barco N.V. - Technology Center, B-8500, Kortrijk, Belgium
[†] Data Science Lab - Ghent University - iMinds, B-9050, Ledeberg-Ghent, Belgium

ABSTRACT

Complex software appliances can be deployed on hardware with limited available computational resources. This computational boundary puts an additional constraint on software applications. This can be an issue for real-time applications with a fixed time constraint such as low delay video encoding. In the context of High Efficiency Video Coding (HEVC), a limited number of publications have focused on controlling the complexity of an HEVC video encoder. In this paper, a technique is proposed to control complexity by deciding between 2Nx2N merge mode and full encoding, at different Coding Unit (CU) depths. The technique is demonstrated in two encoders. The results demonstrate fast convergence to a given complexity threshold, and a limited loss in rate-distortion performance (on average 2.84% Bjøntegaard delta rate for 40% complexity reduction).

Index Terms— Complexity Constrained, Real-time encoding, HEVC

1. INTRODUCTION

When encoding in software, the encoding application itself is usually not the only software process running on the host machine. Other processes can run in parallel, including operating system processes, and even other encoders / decoders. One example could be a cloud service transcoding a number of video streams from one format to the other. When the load on the host machine becomes significant, the software processes may fight for the same resources and as such influence each others processing speed. This can be problematic, particularly for a video encoder that is expected to deliver output at regular time intervals. Although buffering may solve some of these issues, adding a buffer introduces additional delay that may not be desired in a low-delay video application. Therefore, in many cases, the solution consists of carefully fine tuning configuration parameters and/or overprovisioning resources (e.g., selecting one of the more high-end cloud instances). This results in additional effort and financial cost.

Furthermore, when over dimensioning the system for the statically worst case (e.g., assuming that encoding every video stream requires maximum complexity), the configuration may be suboptimal for other cases since in these other cases system resources may not be fully used. When more resources are available, a video encoder may be able to improve quality further, for example. So instead of dimensioning a system for the worst case, it would be interesting to assign a complexity budget to each software process, and have the process reconfigure itself to respect this budget. Such an automatic reconfiguration is particularly useful in cases where the load on the host changes dynamically and/or the same software is expected to flexibly run on multiple hardware platforms (e.g., cloud instances of different providers).

In this paper, we will explore techniques for complexity control in the context of High Efficiency Video Coding (HEVC) [1] video encoding, a computationally complex process. According to [2], by adding new compression tools as well as extending already supported compression tools, HEVC encoding consumes 9 to 502 % more computational complexity compared to its predecessor H.264/AVC [3], depending on the configuration settings. Note that the actual computational complexity also depends on the complexity of the video sequence being encoded which further motivates the need for complexity control.

The state-of-the-art in complexity optimization for HEVC is extensive. However, most algorithms described in the literature do not provide complexity control, but instead focus only on reducing complexity, typically, through early termination of the mode decision or motion search process. For example, Choi et al. [4] proposed to not evaluate split when the best mode is skip. Jaehwan et al. [5] reduce complexity with 35% by deciding early in the encoder evaluation process whether or not to use SKIP mode. Miok et al. [6] propose to terminate motion search early by estimating the RD cost of the merge mode based on the results obtained for motion vector prediction. Zhaoqing et al. [7] propose to use a technique for early merge mode decision based on information from the all-zero block and motion estimation, exploiting also correlation between different CU depths to reduce complexity.

A limited number of researchers have focused explicitly on complexity control, often labeling their work as *complexity constrained encoding* or *complexity scalable encoding*. For example, Kannangara et al. [8] propose techniques for frame-level complexity control of a real-time H.264/AVC encoder. The complexity is controlled by choosing between fast encoding using SKIP mode or full evaluation, based on an es-

timation of the Lagrangian rate-distortion-complexity (RDC) cost for both options. Correa et al. [9] propose a complexity scalability method for HEVC in which CTUs are dynamically constrained by limiting the prediction block (PB) and setting a maximum tree depth for each CTU. Zhao et al. [10] propose a hierarchical complexity allocation scheme for HEVC based on linear programming. While these methods can provide good complexity control, they report relatively high variation of the encoding time over multiple GOPs, which make them less suitable for real-time low delay video encoding.

In this paper, a strategy based on minimizing the RD cost error is proposed. In general, the proposed technique decides between only evaluating 2Nx2N merge mode and full evaluation of CUs by comparing the estimated RD cost error with a dynamic threshold as described in Section 2. The proposed complexity controller can adapt fast to the complexity target with acceptable Bjøntegaard Delta Rate (BD-rate) loss as shown in Section 4. Finally, a conclusion is provided in Section 5.

2. DEFINING A COMPLEXITY THRESHOLD

An HEVC encoder has a wide range of modes / CU structures that can be evaluated to encode each frame. To control complexity, our strategy is therefore to define a threshold that constrains this evaluation process. Since CUs at a higher depth will only be evaluated when their parent CU evaluates to split further, denote Ω the number of CUs (at different depths) that are evaluated for an entire frame, out of a total of M possible CUs. For a CU at index m in this list (with $m \in \{1, ..., M\}$), evaluating the n'th mode (with $n \in \{1, ..., N\}$)[1] results in distortion $D_{m,n}$ and rate $R_{m,n}$, the lagrangian RD cost $J_{m,n}$ is defined with λ as the Lagrange multiplier:

$$J_{m,n} = D_{m,n} + \lambda R_{m,n}, \tag{1}$$

In a typical encoder, only the coding mode with the lowest Lagrange cost will actually be used for encoding. To this end, denote $I_{m,n}$ the optimal RD cost for a CU at index m after evaluating n modes.

The computational complexity required for evaluating mode n for a given CU at index m will be denoted $C_{m,n}$. When evaluating all CU configurations and coding modes, the total frame complexity is given by $\sum_{m=1}^{M} \sum_{n=1}^{N} C_{m,n}$.

2.1. Defining and simplifying the optimization problem

One way to reduce complexity is to not evaluate all encoding modes / depths. To this end, denote α_m the actual number of encoding modes evaluated for a CU at index m ($\alpha_m \in \{1, ..., N\}$). Note that the value of α_m can have an influence on the list of evaluated CUs and therefore change Ω.

[1] N is defined a constant but this value can be different at different depths and/or CU locations. As invalid encoding modes for a specific CU can be ignored, this will not be indicated in the constant N for readability.

Clearly, when only a limited set of encoding modes is evaluated (i.e., with $\alpha_m < N$), there is a possible penalty in terms of RD cost. This penalty will be called the RD cost error E_{m,α_m}:

$$E_{m,\alpha_m} = I_{m,\alpha_m} - I_{m,N}. \tag{2}$$

The RD cost error is zero when the optimal encoding mode is within the first α_m encoding modes, otherwise E_{m,α_m} will be positive. The challenge is now to intelligently define these α_m so that rate and distortion are optimized while the reduced complexity C_R does not exceed a particular complexity threshold C_T. In other words:

$$C_R = \sum_{m=1}^{\Omega} \sum_{n=1}^{\alpha_m} C_{m,n} \tag{3}$$

$$\min_{\alpha_0,...,\alpha_\Omega} \sum_{m=1}^{\Omega} E_{m,\alpha_m} \text{ subject to: } C_R \leq C_T. \tag{4}$$

Solving this constrained optimization problem is complex, particularly since the RD cost (and therefore also I_{m,α_m} and E_{m,α_m}) in general depends on decisions taken in the context of previously coded CU's. If the number of encoding modes to evaluate has been severely restricted for a particular CU, the RD performance for that CU could be relatively low, which increases the RD cost for spatially and temporally neighboring CU's trying to exploit correlation with this CU.

In this paper, α_m is calculated as follows. First, we simplify the problem by ignoring the dependency discussed in the previous paragraph, and assuming that the value for α_m can be decided independently from other CU's. In addition, the goal of our technique is to distribute the Lagrange cost error E_{m,α_m} equally across the frame. To cope with different CU depths, we normalize the Lagrange cost error by dividing it through the number of pixels at a certain depth. As such, denote $\bar{E}_T$ the Lagrange cost error target that we want to achieve, per pixel. Also, denote $\bar{E}_{m,\alpha_m}$ the Lagrange cost error E_{m,α_m} per pixel.

This way, we reformulate Eq. 4 as:

$$\min_{\alpha_0,...,\alpha_\Omega} \sum_{m=1}^{\Omega} |\bar{E}_T - \bar{E}_{m,\alpha_m}| \text{ subject to: } C_R \leq C_T. \tag{5}$$

Based on this equation we build a complexity controller that will vary the target Lagrange error cost $\bar{E}_T$ frame-by-frame in an attempt to match the complexity target C_T. Then, given $\bar{E}_T$ at the start of coding a frame, we estimate $\bar{E}_{m,\alpha_m}$ for each CU and determine when to stop encoding.

3. COMPLEXITY CONTROLLER

In the previous section we assumed that there are N encoding modes. For the implementation in this paper, we simplify

this to two groups of encoding modes for explaining the concept of the complexity controller and the experimental results. Future work includes extending this technique in order to support more encoding modes. The two groups that are used are respectively the 2Nx2N merge mode and another group containing all other encoding modes. The 2Nx2N merge mode has relative low complexity and is used frequently.

3.1. Deciding when to stop encoding

The encoder estimates $\bar{E}_{m,\alpha_m}$ to decide whether or not to continue evaluating other modes, where $\bar{E}_{m,\alpha_m}$ is a normalized version of Eq. 2. While $I_{m,n}$ is available at mode n, $I_{m,N}$ is not available and therefore needs to be estimated. Due to efficient construction of the merge candidate list, there is a high probability that 2Nx2N merge mode is the optimal encoding mode or has low RD cost difference with the optimal mode. So, high correlation is expected. Therefore, in this paper a simple linear model is used to predict $I_{m,N}$. The coefficients of the linear model for depth $\{0, 1, 2, 3\}$ are respectively $\{0.87, 0.92, 0.94, 0.95\}$. A higher coefficient indicates that on average a lower $\bar{E}_{m,\alpha_m}$ is expected if further evaluation of encoding modes is not done. In future work, this model can be extended with higher precision by using more features of the CUs.

3.2. Designing the complexity controller

The goal of the complexity controller is to respect the complexity target C_T. There is a relation between C_R and $\bar{E}_T$. To investigate the relation between C_R and $\bar{E}_T$, an anchor encoder is used to extract I_{m,α_m} and the encoding is re-run with multiple thresholds $\bar{E}_T$. When $\bar{E}_T = 0$, the complexity is reduced only on the CUs where the 2Nx2N merge mode was the optimal mode. When $\bar{E}_T > 0$, different CUs will stop encoding after mode α_m when I_{m,α_m} is below the normalized threshold. The complexity reaches a minimum when all CUs (at depth 0) only evaluate 2Nx2N merge mode. With a prediction model, $\bar{E}_T$ can be defined for a constrained complexity C_T. But, this model is not available for the complexity controller in a real-time implementation. However, C_R and $\bar{E}_T$ are respectively an increasing and a decreasing monotonic function of α_m (see Eq.4). Therefore, the complexity controller can use $\bar{E}_T$ to control the frame complexity C_R, due do the properties of monotonic functions.

This is a typical root finding problem. However, there are additional difficulties compared to conventional root finding algorithms. In this use case, while $\bar{E}_T$ is a constant for a frame, it does change slightly every frame due to the changing complexity of the video or inaccurate measurements of the encoding complexity. Therefore, we need a complexity controller that adapts $\bar{E}_T$ continuously on a frame-by-frame basis. In a conventional root-finding problem the algorithm finds a point closer to the root at each iteration. In this case, only a single iteration can be used each frame and the algorithm can fine-tune this value on the next frame. Therefore, the basic complexity controller proposed in this paper is based on the root-finding bisection method. This proposed method is explained with pseudo-code in Figure 1. If the root is outside the current interval, σ will be doubled if the previous iteration was in the same direction (i.e. σ has the same sign). Doubling σ allows the method to faster search for the correct interval. When the interval with the root is found, σ is divided by -2 (i.e. changing the direction), allowing more precise approximation of the correct value while still being able adapt to local changes in the video or noise on the complexity measurement. Finally, $\bar{E}_T$ is incremented with the current σ value. Note that σ and $equal\_sign$ can be initialized with any value (e.g. respectively σ_0 and $True$), this will only have influence on the iterations needed to find the correct E_F value.

```
σ ← σ_0
equal_sign ← True
function UpdateThreshold(C_T, C_R)
    if (σ ≥ 0) ≠ (C_R > C_T) then
        σ ← σ/ − 2
        equal_sign ← False
    else
        if equal_sign then
            σ ← 2σ
        end if
        equal_sign ← True
    end if
    E_T ← max(0, E_T + σ)
end function
```

Fig. 1: Pseudo code of the proposed complexity controller.

4. EXPERIMENTAL RESULTS

The proposed method is evaluated using sequences from class B described in the common test conditions of HEVC [11] (i.e. Kimono, ParkScene, Cactus, BQTerrace, BasketballDrive). The HM 16.2 reference encoder is used as an anchor. Later, the real-time capability is demonstrated with x265. These experiments focus on low delay settings (i.e. main tier and P slices only), with QP values 22, 27, 32, 37 and open GOP structure.

Table 1 shows the encoding time (ET) per frame for QP=22, for the anchor and proposed method. For the anchor, although encoding settings are identical, there is a significant difference in encoding time (i.e., 42.8%) between Cactus (highest encoding time) and Kimono (lowest encoding time). For the proposed method, a complexity target of 40 seconds per frame was defined. The encoder initializes during the first frames in an attempt to find the RD cost threshold using the bisection method, as described in Section 3. After this initialization phase, the complexity controller adapts the threshold frame-by-frame, and manages to keep computational complexity close to the complexity budget, with an average deviation of 0.57 seconds (1.4%). Reduced complex-

Table 1: Encoding time per frame with QP=22 for the anchor encoder and the complexity constrained encoder with a complexity target of 40 sec/frame. The table provides the average encoding time (ET), PSNR and bitrate.

	Anchor			CCE		
Sequence	ET (sec)	PSNR (dB)	Rate (Mbit/s)	ET (sec)	PSNR (dB)	Rate (Mbit/s)
basketballdrive	65.56	42.69	55.04	40.26	42.59	53.13
bqterrace	73.89	42.93	135.55	39.95	42.62	137.52
cactus	77.09	41.70	92.99	40.10	41.57	94.52
kimono	53.71	43.63	12.69	39.87	43.59	12.49
parkscene	59.82	42.26	20.99	40.09	42.18	20.88

Table 2: Comparison of encoding time (ET) and BD-rate (BDR) for different complexity thresholds with proposed method and two fixed complexity reduction methods (Early skip detection (ESD) and maximum CU depth 2 (MD2)).

Sequence	Proposed Method				Fixed Complexity Reduction			
	80%		40%		ESD [5]		MD2	
	ET	BDR	ET	BDR	ET	BDR	ET	BDR
basketballdrive	0.79	0.07	0.40	6.47	0.74	2.22	0.42	8.98
bqterrace	0.79	1.72	0.39	11.17	0.77	1.56	0.43	6.90
cactus	0.78	0.18	0.39	10.49	0.76	2.00	0.41	12.20
kimono	0.79	0.11	0.40	4.98	0.78	2.37	0.40	3.24
parkscene	0.79	0.00	0.40	13.52	0.69	2.28	0.43	14.18

ity comes at a limited cost, e.g., 0.13 dB PSNR loss and 1.6% bitrate loss for a complexity reduction of 48% for the Cactus sequence.

Fig. 2 shows RD curves for Kimono sequence and different complexity targets (CT). The CT is defined relative to the anchor encoding time at same QP point. For a complexity target of 80%, the BD-rate increase ranges from 0.00% (for ParkScene) to 1.72% (for BqTerrace).

The proposed method for complexity control is compared with a number of fixed complexity methods incorporated in the HM encoder (see Table 2). Using early skip detection (ESD) [5], an average complexity of 74.8% (relative to the anchor) is measured. When only a maximum CU depth of 2 (MD2) is allowed, an average encoding time of 41.8% of the anchor encoding is measured. While our technique provides complexity control, and in general better RD performance compared to the ESD and MD2 methods. Occasionally, the latter methods show better performance, indicating that there is still room for improving our technique further.

The proposed method has been ported to the x265 encoder [12] to demonstrate real-time encoding on constrained systems. When only 2 threads are used on an Intel Xeon CPU E5-2620 v3 running at 2.40GHz, the x265 encoder is only running real-time when configured on a bitrate smaller than 500 kbps. With the proposed method, x265 can encode real-time at all bitrates. The PSNR loss is illustrated for the kimono sequence in Fig. 3. For a bitrate of 2309 kbps, there is 0.75 dB PSNR loss with 44% complexity reduction.

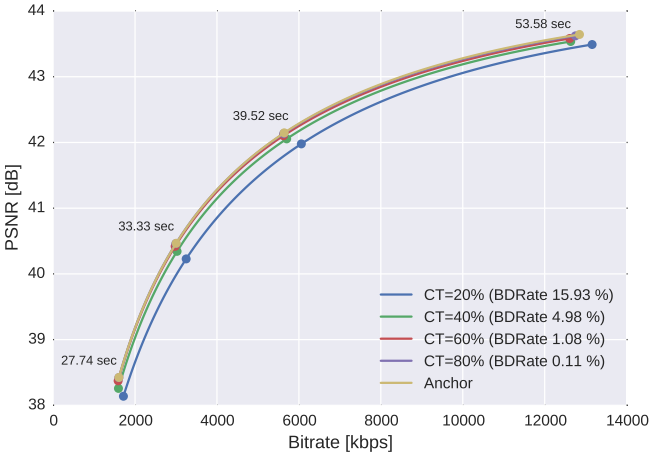

Fig. 2: RD curves for kimono sequence and different complexity thresholds (CT).

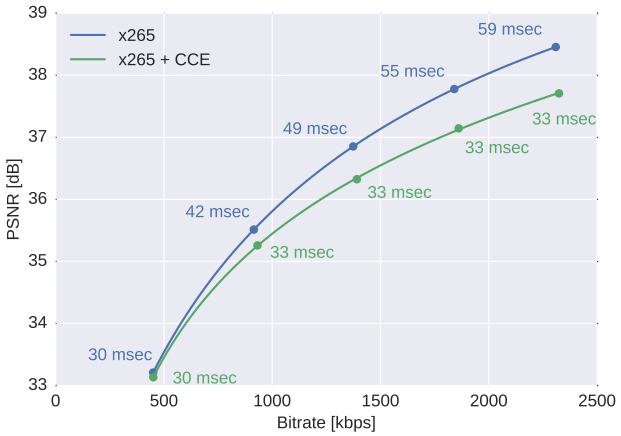

Fig. 3: RD curves with average encoding time per frame for Kimono sequence.

5. CONCLUSION

In this paper we proposed a technique to control the computational complexity of an HEVC encoder. This technique consisted of adaptively deciding between 2Nx2N merge mode or full evaluation, based on a dynamic threshold on the expected RD cost error. Our method operates independently of the host system and the video sequence, and enables constraining the encoding complexity to a given complexity target (after a short initialization phase). Therefore, this technique allows to use software encoding in complexity constrained environments with optimal usage of the available computational complexity (e.g. cloud encoding). The technique is evaluated in two HEVC implementations (e.g. HM and x265) and the results show that the technique is effective, providing fast convergence while incurring only a limited loss in RD performance. Possible extensions on this method have already been identified and are planned as future work.

6. REFERENCES

[1] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, Sep 2012.

[2] G. Correa, P. Assuncao, L. Agostini, and L.A. da Silva Cruz, "Performance and computational complexity assessment of high-efficiency video encoders," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1899–1909, Dec 2012.

[3] T. Wiegand, G.J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, Jul 2003.

[4] K. Choi and E. S. Jang, "Fast coding unit decision method based on coding tree pruning for high efficiency video coding," *Optical Engineering*, vol. 51, no. 3, pp. 030502–1–030502–3, Mar 2012.

[5] K. Jaehwan, Y. Jungyoup, W. Kwanghyun, and J. Byeungwoo, "Early determination of mode decision for HEVC," in *Picture Coding Symposium (PCS), 2012*, May 2012, pp. 449–452.

[6] K. Miok, L. Hyuk-Jae, and L. Nam, "Fast merge mode decision for diamond search in high efficiency video coding," in *Visual Communications and Image Processing (VCIP), 2013*, Nov 2013, pp. 1–6.

[7] P. Zhaoqing, K. Sam, S. Ming-Ting, and L. Jianjun, "Early MERGE mode decision based on motion estimation and hierarchical depth correlation for HEVC," *IEEE Transactions on Broadcasting*, vol. 60, no. 2, pp. 405–412, Jun 2014.

[8] C.S. Kannangara, I.E. Richardson, and A.J. Miller, "Computational complexity management of a real-time H.264/AVC encoder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 9, pp. 1191–1200, Sep 2008.

[9] G. Correa, P. Assuncao, L. Agostini, and L. A. da Silva Cruz, "Complexity scalability for real-time HEVC encoders," *Journal of Real-Time Image Processing*, pp. 1–16, Jan 2014.

[10] Z Tiesong, W Zhou, and K Sam, "Flexible mode selection and complexity allocation in high efficiency video coding," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 7, no. 6, pp. 1135–1144, Dec 2013.

[11] F. Bossen, "Doc. JCTVC-L1100: Common test conditions and software reference configurations," Tech. Rep., JCTVC, Geneva, Switzerland, Jan 2013.

[12] MulticoreWare Inc., "x265 hevc encoder / h.265 video codec," http://www.x265.org, 2016.